

Topological code Autotune

Austin G. Fowler¹, Adam C. Whiteside¹, Angus L. McInnes¹, Alimohammad Rabbani²

¹*Centre for Quantum Computation and Communication Technology,*

School of Physics, The University of Melbourne, Victoria 3010, Australia,

²*Sharif University of Technology, Department of Computer Engineering, Tehran, Iran*

(Dated: February 29, 2012)

Tailoring a fault-tolerant quantum error correction scheme to a specific physical architecture can be a laborious task. We describe a tool Autotune capable of analyzing and optimizing the classical processing for an arbitrary 2-D qubit architecture making use of arbitrary circuits implementing either the surface code or progressively generated slices of a 3-D topological cluster state with arbitrary stochastic error models for each quantum gate. Autotune is designed to facilitate precise study of the performance of real hardware running topological quantum error correction.

Large-scale fault-tolerant simulations of the surface code indicate gate error rates between 0.2% and 0.5% are sufficiently low to enable practical overhead, high reliability quantum computation [1]. This is tantalisingly close to experimentally achieved two-qubit gate error rates of 2% [2], the best achieved to date in a system with the potential to implement the required 2-D array of qubits. This motivates the serious study of mapping topological quantum error correction (TQEC) schemes to physical hardware to enable realistic engineering trade-offs to be determined and optimizations found. We would very much like to collaborate with any experimentalist with a potentially 2-D qubit system and an interest in making use of TQEC. Given the transversely invariant nature of TQEC, experiments involving as few as two qubits can be sufficient to determine whether a system could successfully implement TQEC.

This paper is not self-contained. We assume the reader has basic familiarity with surface code quantum computation and topological cluster states [3–6]. Our primary focus will be explaining how our tool Autotune works and is used.

The discussion is organized as follows. In Section I, we describe how error models are defined for each quantum gate and the current range of phenomena they can encompass. In Section II, we describe how errors are tracked during simulation. In Section III, the concept of a set of measurements is introduced, with a -1 product of measurements in a set indicating a detection event, defined in Section IV. Away from boundaries, every single error leads to a pair of detection events. We store this information in detail in a nest (Section V). This detailed information can be distilled on the fly into a lattice (Section VI), however this is somewhat computationally expensive (a large constant overhead) and so we describe how to distil this information once during a bootup phase into a recipe (Section VII) that enables high-speed construction of the appropriate lattice for matching without the expensive tracking of errors. Having described Autotune, we give examples of its use to analyze the standard surface code (Section VIII) and progressively generated 3-D topological cluster states (Section IX). Section X concludes with a discussion of our planned future work.

I. ERROR MODELS

Autotune is capable of handling any error model with outcomes that can be described by a single integer e per qubit. For example, $I = 0$, $X = 1$, $Z = 2$, $Y = 3$, leaked to a non-computational state = 4, qubit lost = 5. Only single-qubit and two-qubit gate error models are currently supported, however this could easily be extended. The user can specify how e is transformed by each gate, for example controlling whether a CNOT between a leaked qubit and a non-leaked qubit results in two leaked qubits, or no effect on the non-leaked qubit, or any other effect describable by single integers. It must be possible to deterministically say whether each error e contains X and/or Z or neither. Y contains both, I neither, and a leaked qubit may look like either or neither depending on the underlying physics.

An example of a Pauli channel CNOT error model file is shown below.

$$\begin{array}{rcl}
 2 & & \\
 1.0 & & \\
 6 & & \\
 81 & 0 & 3 \\
 14 & 1 & 0 \\
 13 & 2 & 1 \\
 28 & 2 & 2 \\
 78 & 3 & 0 \\
 30 & 3 & 2 \\
 1 & &
 \end{array} \tag{1}$$

The first line states the number of qubits n_q the gate is applied to. The second line states the value x the relative strengths s_i of the various errors should be normalized to sum to. This makes it easy to handle different gates with different overall probabilities of error, e.g. $x_{1q} = x_{2q}/10$. The third line states the number of different errors n_e in the model. The next n_e lines contain $n_q + 1$ integers specifying the relative strength of that error and the value of e to apply to each qubit. The user can specify exactly how different errors combine $e_1 e_2 = e_3$. Each time a quantum gate is called, it is passed an error rate p and an

error of any kind is applied with probability px . Error i is then applied with mutually exclusive probability $s_i/\sum s_i$. The final line is the gate duration in arbitrary units.

II. ERROR TRACKING

Error models are not only used to generate stochastic errors, we also deterministically track a representative set of errors. Specifically, given an arbitrary single-qubit error model we sum the relative probabilities of errors that contain X and those that contain Z and track these two non mutually exclusive errors only. Given an arbitrary two-qubit error model we track the non mutually exclusive errors IX , XI , XX , IZ , ZI and ZZ . This is done simply to keep the number of tracked errors under control. Transforming the CNOT error model in Section I in this manner yields

$$\begin{aligned} p_{IX} &= 0.385, \\ p_{XI} &= 0.5, \\ p_{XX} &= 0, \\ p_{IZ} &= 0.332, \\ p_{ZI} &= 0.373, \\ p_{ZZ} &= 0.238. \end{aligned} \quad (2)$$

Every time a gate is applied, all nonzero relative probability pure X and pure Z errors are generated and added to the list of errors on each qubit touched by the gate. Relative probabilities are stored in each error scaled by the p passed to the gate. Each error is given a unique label. Multiple-qubit errors are represented by single-qubit errors on each qubit each with the same label. If one executes a long sequence of unitary gates, the number of errors per qubit that need to be tracked will grow without bound. Each unitary gate transforms all errors present on all of the qubits it touches. An S gate will transform X errors into Y errors and vice versa. Multiple-qubit gates can create new propagated errors which will have the same label as the original and can combine or cancel multiple errors on a single qubit with the same label. For example, $\text{CNOT}(q_1, q_2, p = 0.01)$ applied to qubits

$$\begin{aligned} q_1 &\rightarrow (X, 0.005, 0) \\ &\hookrightarrow (Z, 0.00238, 1) \end{aligned} \quad (3)$$

$$q_2 \rightarrow (Y, 0.00238, 1) \quad (4)$$

where (A, p_{sr}, L) represents the error type, scaled relative probability and label and the arrows represent a linked list, will result in

$$\begin{aligned} q_1 &\rightarrow (X, 0.005, 0) \\ &\hookrightarrow (X, 0.005, 3) \\ &\hookrightarrow (Z, 0.00373, 5) \\ &\hookrightarrow (Z, 0.00238, 6) \end{aligned} \quad (5)$$

$$\begin{aligned} q_2 &\rightarrow (X, 0.005, 0) \\ &\hookrightarrow (Y, 0.00238, 1) \\ &\hookrightarrow (X, 0.00385, 2) \\ &\hookrightarrow (Z, 0.00332, 4) \\ &\hookrightarrow (Z, 0.00238, 6) \end{aligned} \quad (6)$$

III. SETS OF MEASUREMENTS

Autotune currently supports only single-qubit measurements in the X and Z bases. M_X applied to eq. 6 will create a measurement

$$\begin{aligned} m &\rightarrow (Z, 0.00238, 1) \\ &\hookrightarrow (Z, 0.00332, 4) \\ &\hookrightarrow (Z, 0.00238, 6) \end{aligned} \quad (7)$$

The X errors and X components of Y errors have been removed. All errors will be removed from the qubit. To use the qubit again it must be explicitly initialized. If the qubit does not need to be used immediately after being measured, Autotune provides a dead gate that advances the qubit in time but does not generate or track any errors. This models incoherent evolution.

Every measurement is associated with either two sets or a single set and a boundary. A set of measurements has the property that the product of the measurement results (+1 or -1) indicates whether a chain of errors has ended nearby. In the standard surface code, sets contain consecutive pairs of syndrome qubit measurements. In a 3-D topological cluster state, sets contain the measurements on the faces of individual primal and dual cells. See Fig. 1. Sets must be specified by the user.

Sets can also be associated with boundaries. The bottom layer of sets in Fig. 1a is associated with the primal initial time boundary. In the second layer of sets, sets in the left row are associated with the left primal boundary, those in the right row are associated with the right primal boundary. We use the terminology primal/dual instead of rough/smooth as used in [5] to ensure uniform terminology when discussing both the surface code and 3-D topological cluster states. Note that the middle row of sets in the second layer is not associated with any boundary. The association of sets with boundaries is currently manually user specified.

IV. DETECTION EVENTS

When all measurements in a set have been performed, further processing is triggered. A measurement may contain many errors. A set may contain many measurements. Autotune determines which errors with the same label appear an odd number of times. For each such label a detection event is generated. Fig. 2a contains an example of a single error leading to a pair of detection

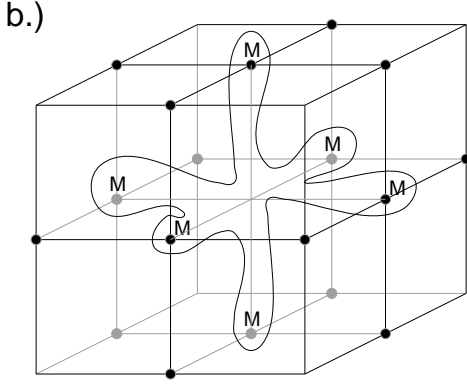
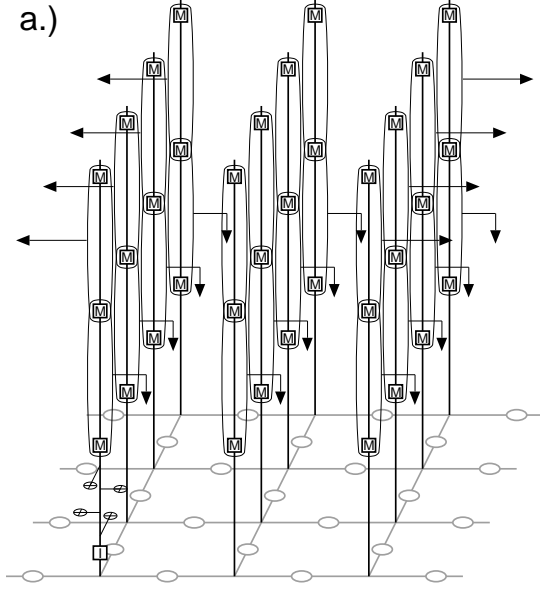


FIG. 1: a.) 2-D surface code (grey). Time runs vertically. In the front left corner, the transversely invariant initialization, 4 CNOT, then measurement pattern is shown. For clarity, only the measurement gate of this pattern is shown elsewhere in space-time. Long ovals encircle pairs of measurements and represent sets. Arrows indicate associations with boundaries. The bottom layer of sets is associated with the initial time boundary. The second layer left and right rows of sets are associated with the left and right spatial boundaries respectively. b.) A single 3-D topological cluster state cell. Sets contain six measurements away from boundaries of the lattice.

events. Detection events are stored in a hash table to enable one to quickly determine whether a detection event with a given label has already been generated. Pairs of detection events immediately trigger the creation of sticks, described in the following Section.

Errors near a boundary can lead to single detection events (Fig. 2b). Such single detection events must gen-

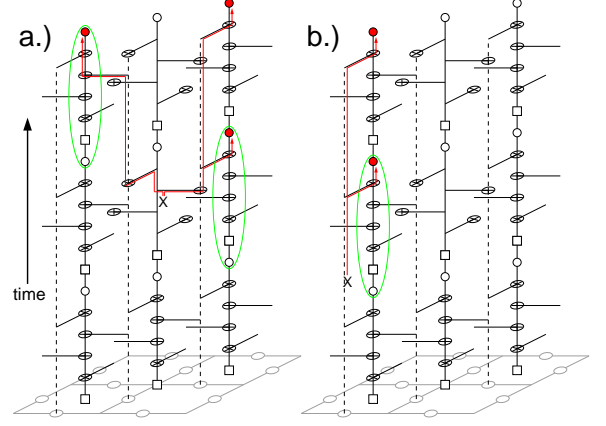


FIG. 2: 2-D surface code (grey). Time runs vertically. Squares represent initialization to $|0\rangle$, circles represent Z basis measurement. a.) A single error leading to a pair of detection events. b.) An error leading to a single detection event due to proximity to a boundary of the lattice.

erate sticks leading to the nearby boundary. One must decide with care when to conclude that a detection event is unique and that no matching detection event will be generated in the future. To deal with this, we define a measure of error detection progress big_t to increment only when every stabilizer of the code has been measured at least once. An example of an error leading to detection events two big_t in the future is shown in Fig. 3. This is the maximum delay possible. Detection events three big_t in the past that are unique are guaranteed to remain so.

Detection events can be primal or dual depending whether they are associated with primal or dual sets which are in turn associated with primal or dual stabilizer measurements. It is a good idea to design stabilizer measurement circuits with the property that single errors do not lead to the creation of more than one pair of primal and one pair of dual detection events. This ensures that minimum weight perfect matching [7, 8] is well-suited to correcting errors generated during the execution of the circuits.

V. NESTS OF BALLS AND STICKS

When a set is processed to generate detection events, it is said to be finalized. At this point in time, a ball is generated and associated with the finalized set. This ball represents a location in space-time. A pair of detection events leads to the creation of a stick between its associated balls. A stick represents a potential connection between a pair of space-time locations arising from a single error. Many single errors can lead to the same stick [9]. The probability of a given stick, given a list of errors leading to it each with its own probability (e_i, p_i) ,

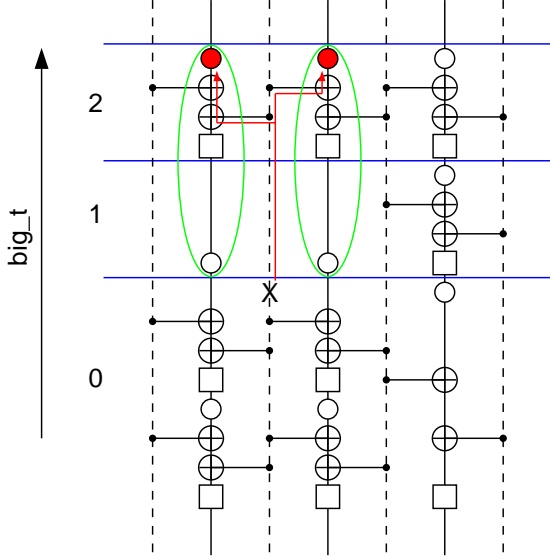


FIG. 3: Example of a single error leading to two detection events two big_t in the future. A detection event has a big_t equal to that of its latest measurement. big_t increments after the every stabilizers has been measured at least once (circles represent measurement, squares represent initialisation).

is therefore, to first order

$$p_{\text{stick}} = \sum_i p_i \prod_{j \neq i} (1 - p_j). \quad (8)$$

We have a simple Blender based visualization tool for nests, a screen shot of which can be found in Fig. 4. A nest contains full tracking information of which gate led to which collection of errors and which of those errors led to which sticks. This is highly useful, however computationally cumbersome. Autotune does delete all errors, measurements, sets, detection events, balls and sticks when they are no longer required, which keeps total memory required finite, however it remains challenging to generate the nest fast enough to obtain good statistics in simulations, let alone keep pace with a real quantum computer.

VI. LATTICES OF DOTS AND LINES

Motivated by the difficulty of rapidly generating nests, we first cut down the data stored therein to the minimum required by the minimum weight perfect matching algorithm. For every ball in a nest we create a dot, which again simply represents a space-time location. For every stick, we create a line. Lines connect the dots corresponding to the balls the stick connected, however they contain just one number, the weight $w = -\ln(p_{\text{stick}})$.

The lattice is kept for much longer than the nest as with low probability the matching algorithm can require

all prior matching history to correctly match the latest data. A sufficiently long history must be kept to ensure the probability of requiring more is negligible. This is possible as the probability of requiring additional data in the past decreases exponentially with kept history size. The simplicity of the lattice keeps the memory required low.

VII. RECIPES OF OFFSETS, BLOCKS AND LAYERS

The visualized nest in Fig. 4 has a great deal of regularity in its structure. This can be exploited to enable direct generation of lattices, avoiding the need to generate expensive nests on the fly. During a boot-up phase, Autotune analyzes each new stick and stores new unique sticks as an offset that contains only the geometric information associated with the stick — just enough to create a line. Furthermore, each new ball is analyzed to determine if it corresponds to a new pattern of offsets. Such unique patterns are stored as blocks of offsets. Finally, each round of error detection is analyzed with structurally unique rounds stored as layers of blocks. All data is then stored in a recipe that contains all necessary information to rapidly create any part of the lattice.

Currently, Autotune is capable of analyzing either error detection circuitry that eventually leads to identical repeated layers or a finite number of rounds of error detection that can have any structure whatsoever. The former could easily be extended to a finite number of cyclically repeated layers, however it remains unclear whether one could avoid generating full nests when simulating probabilistic error detection in which all syndrome measurements can take a randomly variable amount of time.

VIII. EXAMPLE: THE SURFACE CODE

Consider the stabilizer measurement circuits shown in Fig. 5. These circuits are highly physical, with initialization to $|0\rangle$ and Z basis measurement only, and specifically no $|+\rangle$ initialization or X basis measurement as was used in [1, 10]. Note the use of incoherent evolution dead gates D to ensure that Z -stabilizer measurement takes as long as X -stabilizer measurement. We do not assume that all gates take equal amounts of time. Gate D is trivially made to take the same time as H , however all other types of gates are allowed to take arbitrary amounts of time. This implies we need four different duration identity gates — I_{init} , I_H , I_{meas} and I_{CNOT} . These different identity gates are applied to the data qubits at appropriate times while syndrome qubits are undergoing the corresponding operations.

Assuming equal probability p depolarizing noise on all unitary gates and incorrect initialization and measurement with probability p , the low distance performance is

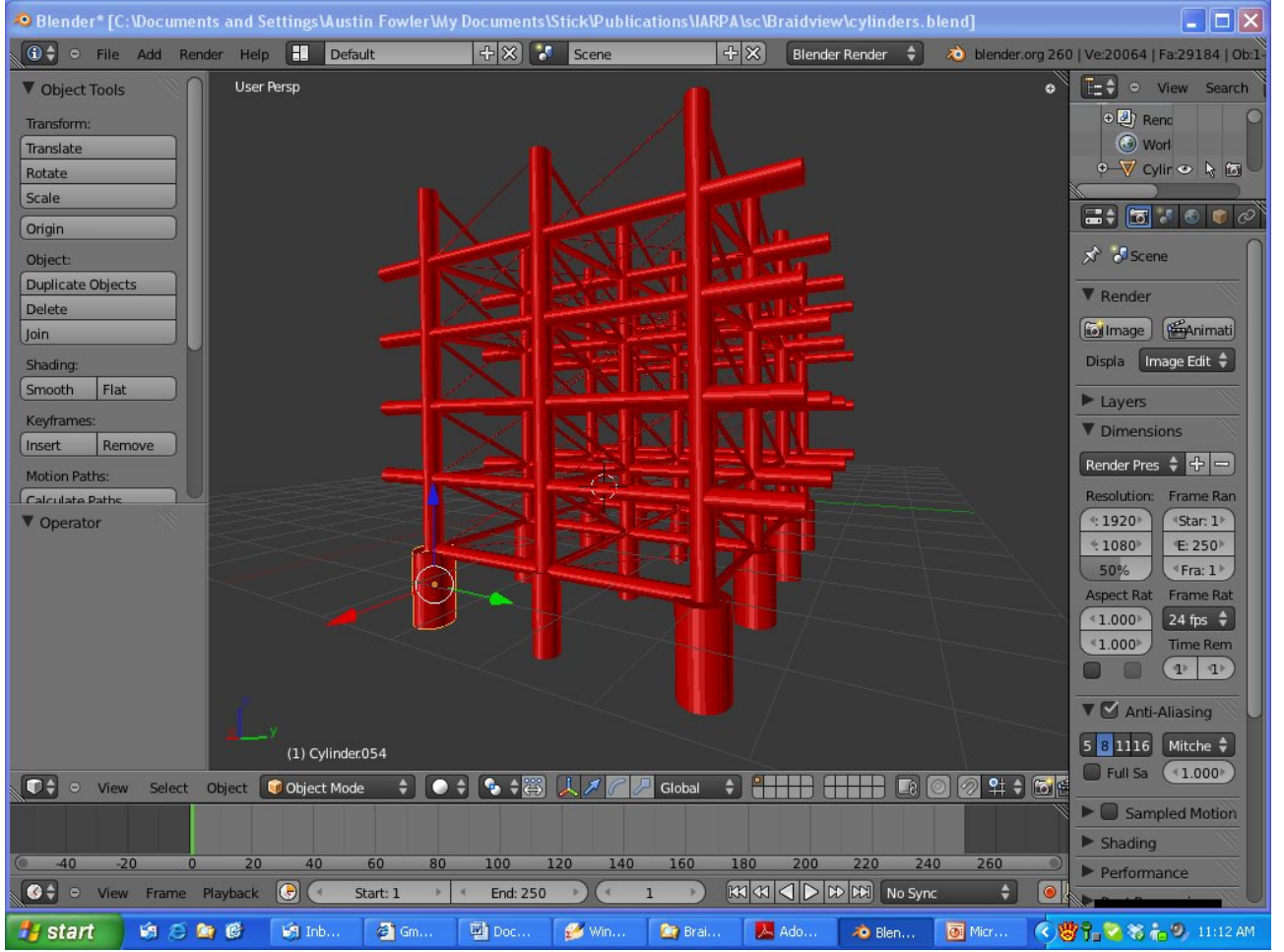


FIG. 4: 3-D Blender model of the probability of sticks. Thicker cylinders represent more probable sticks. Time runs vertically, the $(x, y) = (0, 0)$ position is the back corner.

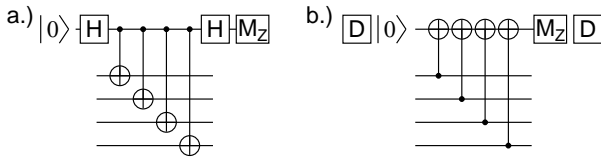


FIG. 5: Physically reasonable syndrome measurement circuits. D gates represent incoherent evolution.

shown in Fig. 6. We expect a significantly lower threshold error rate than that observed in [1] due to the additional gates present in the syndrome measurement circuits. The high distance performance about the threshold error rate is shown in Fig. 7. We observe a threshold error rate of approximately 0.57%.

IX. EXAMPLE: TOPOLOGICAL CLUSTER STATES

A 3-D topological cluster state can be progressively generated using two layers of qubits and a repeating 6-step cycle of measurements and interactions. This is described in detail in Fig. 8. A single layer of qubits is sufficient provided one can have some overlapping but still nearest neighbor interactions (Fig. 9). Sets require more care to construct as they now contain up to six measurements, one on each face of each primal and dual cell. The generic form of the primal and dual cell structure of the cluster state we will simulate is shown in Fig. 10.

We assume the same error model as that used for the surface code in the previous Section, however we permit the use of initialization to $|+\rangle$ and M_X . Initialization to $|+\rangle$ produces $|-\rangle$ with probability p . M_X reports the wrong eigenstate with probability p . Measurement must be followed by initialization. The dual nest resulting from the described gate sequence and error model for a distance 4 topological cluster state is shown in Fig. 11.

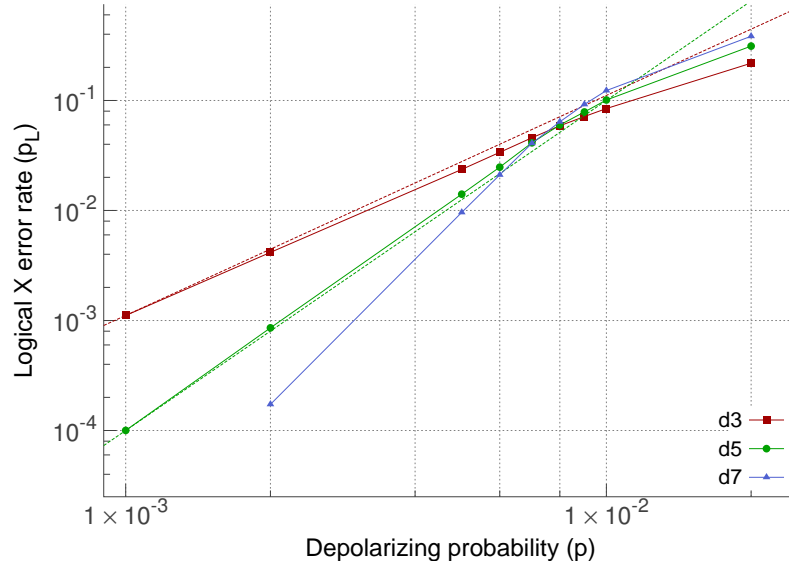


FIG. 6: Probability of logical X error per round of error correction for different square surface code distances d . Quadratic and cubic asymptotic curves are shown for distances 3 and 5 respectively.

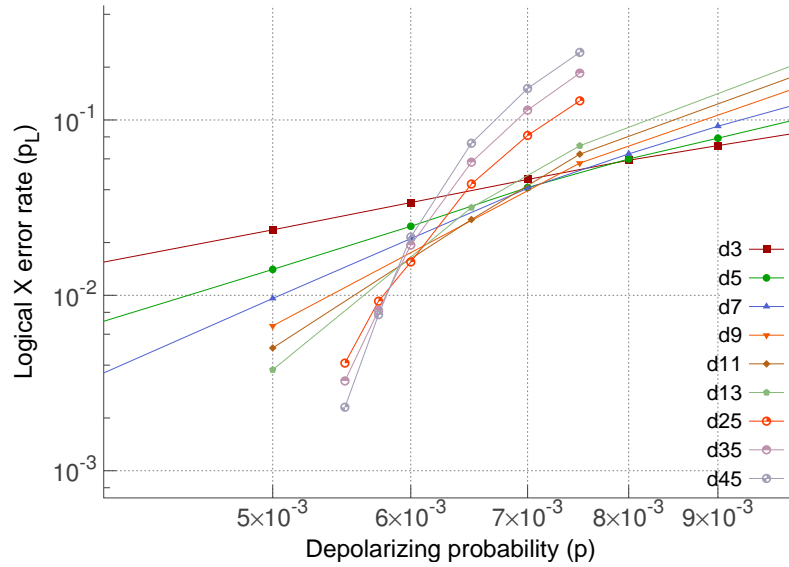


FIG. 7: Probability of logical X error per round of error correction for large distances around the threshold error rate of approximately 0.57%

Note that only one type of logical error is now possible as we have a single unbroken U-shaped primal boundary for all time and the only nontrivial errors connect the two opposing dual boundaries.

As was done for the surface code, for low error rates we can compare the actual performance to the expected asymptotic behavior. A logical error in a distance 3 code requires a minimum of two physical errors to occur, so

for low error rates a distance 3 code is expected to have an error rate quadratic in the depolarizing probability p . Similarly, distance 5 codes should converge to a cubic curve and distance 7 to a quartic curve. It can be seen from Fig. 12 that our results converge well to the expected single-term polynomials.

In addition to the asymptotic behavior, the threshold error rate is of interest. Fig. 13 shows the behavior

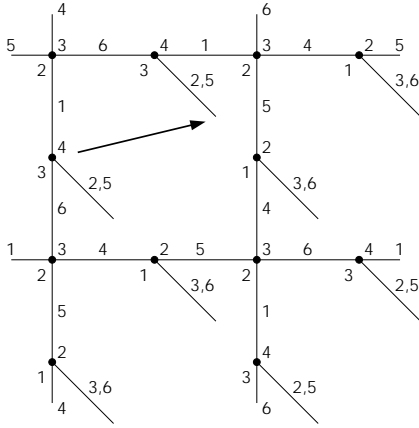


FIG. 8: A unit cell of the interaction pattern of a double layer of qubits generating a 3-D cluster state. Small black dots correspond to qubits. Most of the qubits in the second layer have been omitted for clarity. The numbers at the top right and bottom left of each qubit indicate the time steps in which it is initialized and measured, respectively. Each line corresponds to a C_Z gate and is labeled with the time step(s) in which this gate is applied. The other layer comprises an identical ordering of qubits and gates, but shifted in time by three steps (such that a gate labeled 2 is executed in time step 5 and vice versa, etc.) and shifted north, east and up as indicated by the arrow. Upward C_Z gates in the first layer become downward after shifting.

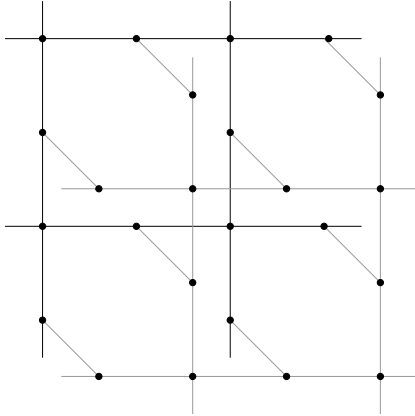


FIG. 9: A single-layer implementation of the two layers used to progressively generating a 3-D topological cluster state. Note that while some interactions cross, they are still technically nearest neighbor.

around threshold for the error model described above. The observed threshold area of approximately 0.58% is lower than the threshold obtained in [4] due to the more severe error model. Fig. 14 is a repeat of the same calculation using depolarizing noise after initialization and before measurement leading to an approximate threshold error rate of 0.66%, in good agreement with the 0.67% value in eq. 23 of [4].

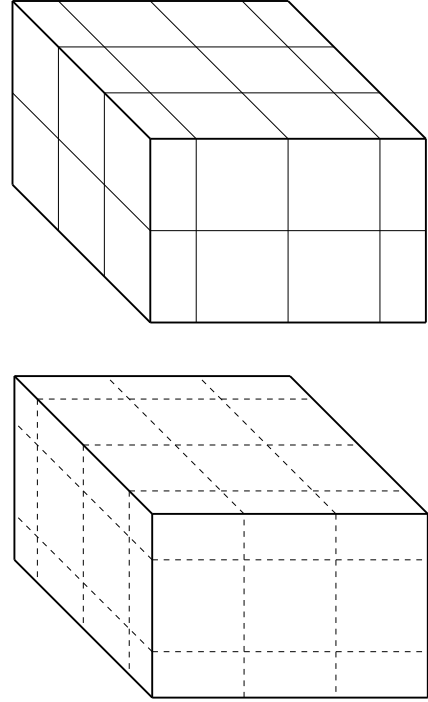


FIG. 10: Primal cells (top) and dual cells (bottom) in a distance 4 example of the topological 3-D cluster states we simulate. Dual boundaries are present on the left and right sides and a single U-shaped primal boundary covers the bottom, front and back. Time runs vertically. The top temporal surface should not be considered a boundary as the cluster state is being extended in this direction.

X. CONCLUSION

We have described a code library Autotune that is capable of handling in a natural manner both fully fault-tolerant surface codes and 3-D topological cluster states. Generality is achieved through the definition of sets of measurements which are used to detect error chain endpoints. Arbitrary syndrome measurement circuits are supported along with arbitrary stochastic error models for each gate. The code is highly efficient, with runtime comparable to that reported in [11].

In future work, we plan to take correlations between errors into account, parallelize the core matching engine, and develop the capability to analyze complex fault-tolerant circuits consisting of many braided defects.

XI. ACKNOWLEDGEMENTS

This research was conducted by the Australian Research Council Centre of Excellence for Quantum Computation and Communication Technology (project number CE110001027). Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Depart-

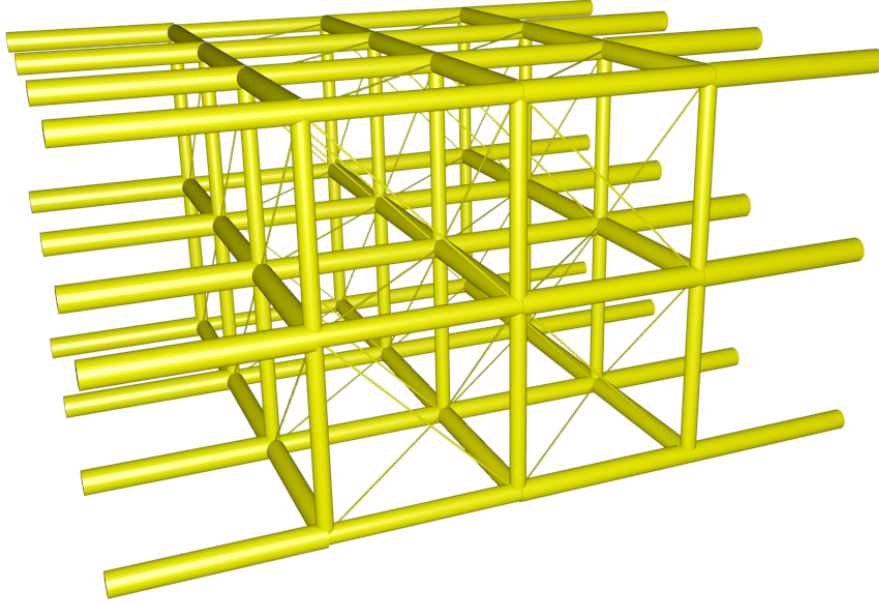


FIG. 11: The dual nest associated with the cluster state of Fig. 10.

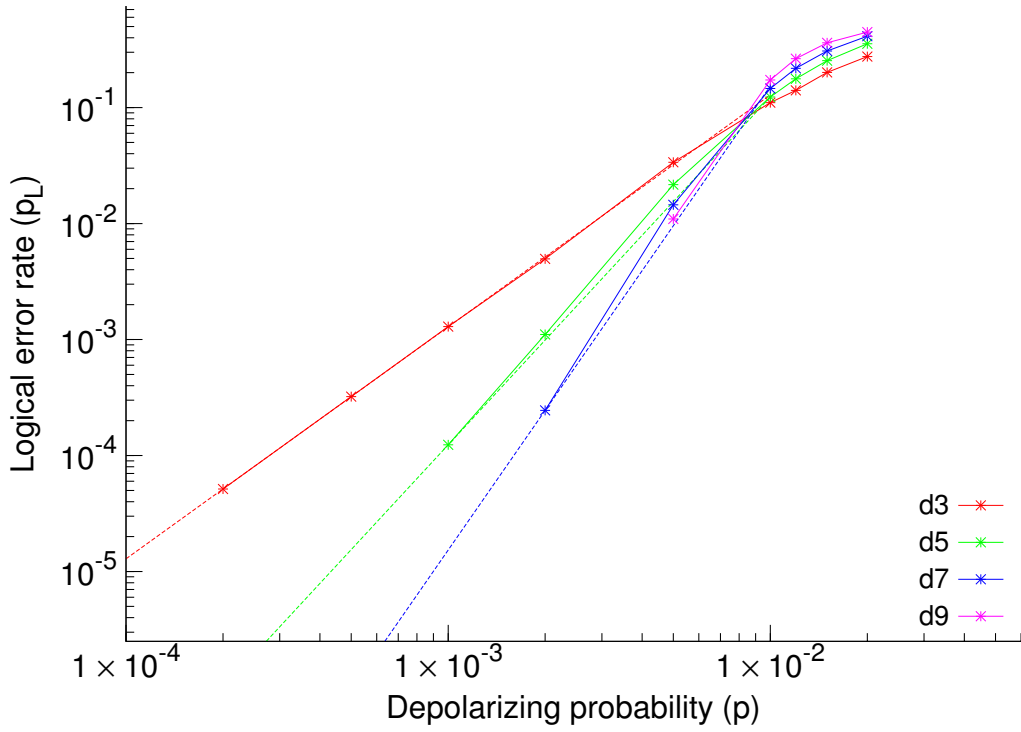


FIG. 12: Logical error rate per layer of cluster state cells for distances 3, 5, 7 and 9. Dashed lines are polynomials showing the expected quadratic, cubic and quartic single-term polynomial asymptotic behaviors.

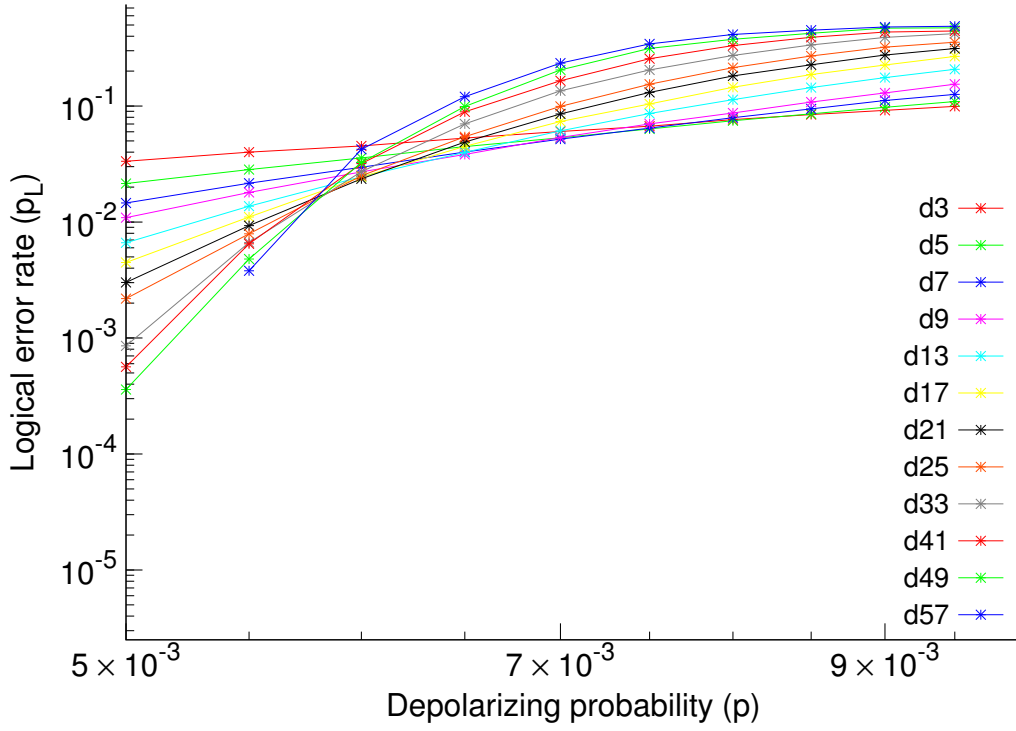


FIG. 13: Logical error rate per layer of cluster state cells for a large range of distances around the threshold error rate. Initialization and measurement fail with probability p . The threshold error rate for this error model is approximately 0.58%.

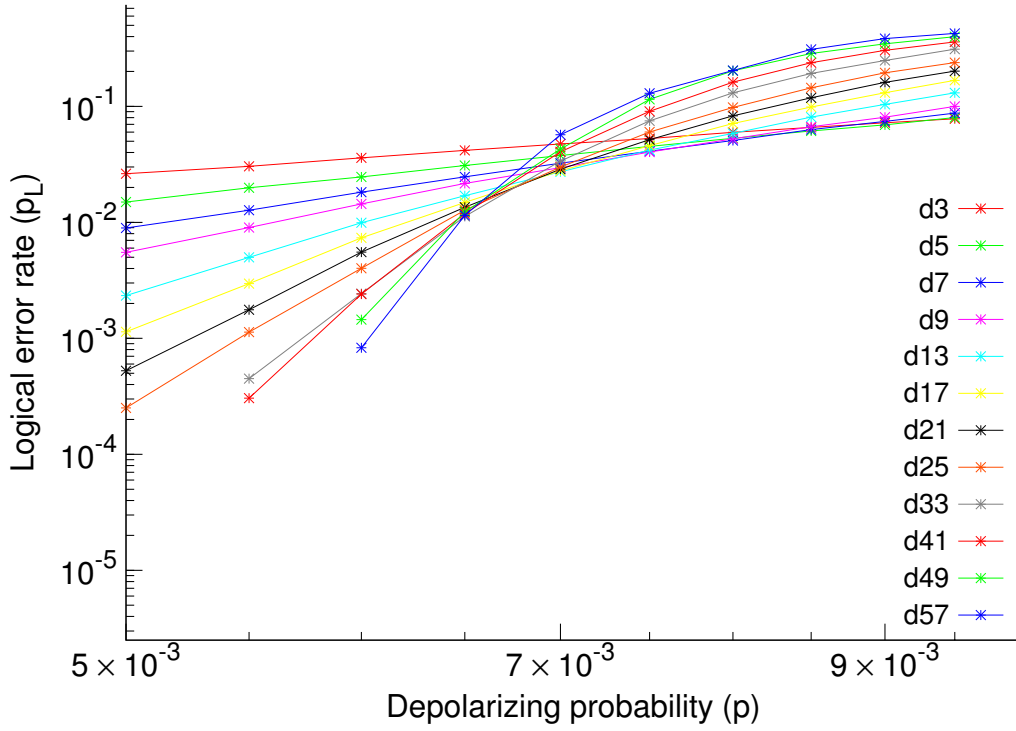


FIG. 14: Logical error rate per layer of cluster state cells for a large range of distances around the threshold error rate. Initialization and measurement suffer depolarizing noise with probability p . The threshold error rate for this error model is approximately 0.66%.

ment of Interior National Business Center contract number D11PC20166. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein

are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

-
- [1] A. G. Fowler, A. C. Whiteside, and L. C. L. Hollenberg, arXiv:1110.5133 (2011).
 - [2] T. Monz, P. Schindler, J. T. Barreiro, M. Chwalla, D. Nigg, W. A. Coish, M. Harlander, W. Hänsel, M. Hennrich, and R. Blatt, Phys. Rev. Lett. **106**, 130506 (2011), arXiv:1009.6126.
 - [3] R. Raussendorf and J. Harrington, Phys. Rev. Lett. **98**, 190504 (2007), quant-ph/0610082.
 - [4] R. Raussendorf, J. Harrington, and K. Goyal, New J. Phys. **9**, 199 (2007), quant-ph/0703143.
 - [5] A. G. Fowler, A. M. Stephens, and P. Groszkowski, Phys. Rev. A **80**, 052312 (2009), arXiv:0803.0272.
 - [6] A. G. Fowler and K. Goyal, Quant. Info. Comput. **9**, 721 (2009), arXiv:0805.3202.
 - [7] J. Edmonds, Canad. J. Math. **17**, 449 (1965).
 - [8] J. Edmonds, J. Res. Nat. Bur. Standards **69B**, 125 (1965).
 - [9] D. S. Wang, A. G. Fowler, and L. C. L. Hollenberg, arXiv:1009.3686 (2010).
 - [10] D. S. Wang, A. G. Fowler, and L. C. L. Hollenberg, Phys. Rev. A **83**, 020302(R) (2011), arXiv:1009.3686.
 - [11] A. G. Fowler, A. C. Whiteside, and L. C. L. Hollenberg, arXiv:1202.5602 (2012).